

A hierarchy of behavioral equivalences in the π -calculus with noisy channels

Yongzhi Cao^{1,2,*}

¹*Institute of Software, School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China*

²*Key Laboratory of High Confidence Software Technologies (Peking University)
Ministry of Education, China
E-mail: caoyz@pku.edu.cn*

Abstract

The π -calculus is a process algebra where agents interact by sending communication links to each other via noiseless communication channels. Taking into account the reality of noisy channels, an extension of the π -calculus, called the π_N -calculus, has been introduced recently. In this paper, we present an early transitional semantics of the π_N -calculus, which is not a directly translated version of the late semantics of π_N , and then extend six kinds of behavioral equivalences consisting of reduction bisimilarity, barbed bisimilarity, barbed equivalence, barbed congruence, bisimilarity, and full bisimilarity into the π_N -calculus. Such behavioral equivalences are cast in a hierarchy, which is helpful to verify behavioral equivalence of two agents. In particular, we show that due to the noisy nature of channels, the coincidence of bisimilarity and barbed equivalence, as well as the coincidence of full bisimilarity and barbed congruence, in the π -calculus does not hold in π_N .

Keywords: π -calculus, π -calculus with noisy channels, barbed equivalence, barbed congruence, bisimilarity.

1 Introduction

The need for formal methods in the specification of concurrent systems has increasingly become well accepted. Particular interest has been devoted to Petri nets [36, 39], CSP [22, 23], ACP [10], CCS [28, 29], and the π -calculus [32]. The last one due to Milner *et al.* was developed in the late 1980s with the goal of analyzing the behavior of mobile systems, i.e., systems whose communication topology can change dynamically, and it turns out to be the unique one among the aforementioned calculi that can express mobility directly. The π -calculus has its roots in CCS, namely CCS with mobility, introduced by Engberg and Nielsen [16], while the capacity of dynamic reconfiguration of logical communication structure gives the π -calculus a much greater expressiveness than CCS.

In the π -calculus, all distinctions between variables and constants are removed, communication channels are identified by names, and computation is represented purely as the communication of names across channels. The transfer of a name between two agents (processes) is therefore the fundamental computational step in the π -calculus.

*Supported in part by the National Foundation of Natural Sciences of China under Grants 60505011, 60496321, and 60736011.

The basic (monadic) π -calculus allows only communication of channel names. There are two extensions of such a communication capability: One is the polyadic π -calculus [30] that supports communication of tuples, needed to model passing of complex messages; the other is the higher-order π -calculus [40] that supports the communication of process abstractions, needed for modeling software composition within the calculus itself. Interestingly, both of them can be faithfully translated into the basic π -calculus.

As we see, communication is a key ingredient of the π -calculus. It is worth noting that all communication channels in the π -calculus are implicitly presupposed to be noiseless. This means that in a communication along such a channel, the receiver will always get exactly what the sender delivers. However, it is usually not the case in the real world, where communication channels are often not completely reliable. Recently, Ying [54] took into account the noise of channels, an idea advocated in his earlier paper [51], and proposed a new version of the π -calculus, called the π_N -calculus. Such a calculus has the same syntax as that of the π -calculus. The new feature of π_N arises from a fundamental assumption: Communication channels in π_N may be noisy. This means that what is received at the receiving end of a channel may be different from what was emitted.

According to a basic idea of Shannon’s information theory [45] that noise can be described in a statistic way, the noisy channels in π_N was formalized in [54] as follows: Firstly, like in π , all (noisy and noiseless) communication channels are identified by names. Secondly, to describe noise, every pair of (channel) names x and y is associated with a probability distribution $p_x(\cdot|y)$ over the output alphabet (here it is just the set of names), where for any name z , $p_x(z|y)$ indicates the probability that z is received from channel x when y is sent along it. Finally, based on the probability information arising from the noisy channels, a late probabilistic transitional semantics of π_N is presented. The essential difference between this semantics and that of π is mainly caused by the actions performed by an output agent $\bar{x}y.P$. In π , this agent has a single capability of sending y via channel x , expressed as the transition $\bar{x}y.P \xrightarrow{\bar{x}y} P$, which implies that the same name y will be received at the receiving end of the channel. However, because of noise, in the π_N -calculus the corresponding transition would be $\bar{x}y.P \xrightarrow{\bar{x}z}_{p_x(z|y)} P$. This probabilistic transition indicates that although the name intentionally sent by the agent is y , the name at the receiving end of the channel x may be the name z , different from y , with the probability $p_x(z|y)$. We refer the reader to Section 1.2 in [54] for a comparison between this model of noisy channels and the existing literature [1, 2, 3, 4, 5, 6, 8, 9, 20, 25, 27, 38] including some works about other formal models with unreliable communication channels.

It is well known that behavioral equivalences play a very important role in process algebras because they provide a formal description that one system implements another. Two agents are deemed equivalent when they “have the same behavior” for some suitable notion of behavior. In terms of the π -calculus, various behavioral equivalences have been studied extensively; examples are [7, 11, 12, 18, 32, 34, 37, 40, 41, 42]. In [54], some concepts of approximate bisimilarity and equivalence in CCS [50, 52, 53] were generalized into the π_N -calculus; such behavioral equivalences involve quantitative information—probability, since the agent in π_N is represented by a probabilistic transition system. To our knowledge, except for this work there are no probabilistic versions of behavioral equivalences in the π -calculus and its variants, although probabilistic extensions of the π -calculus [19, 20, 46] were introduced.

The purpose of this paper is to extend some classical behavioral equivalences related

to (strong) barbed equivalence and (strong) bisimilarity into π_N and cast them in a hierarchy. Following the model of noisy channels in [54], we develop an early transitional semantics of the π_N -calculus which makes the study of behavioral equivalences somewhat simpler. It is worthy of note, however, that unlike in the π -calculus, this semantics is not a directly translated version of the late semantics of π_N in [54]. Surprisingly, we have found that not all bound names in π_N are compatible with alpha-conversion when we remove the strong assumption in [54] that free names and bound names are distinct. As a result, we have to add a rule for inputting bound names to the corresponding early semantics of π . In addition, to handle transitions of π_N well, we group the transitions according to their sources.

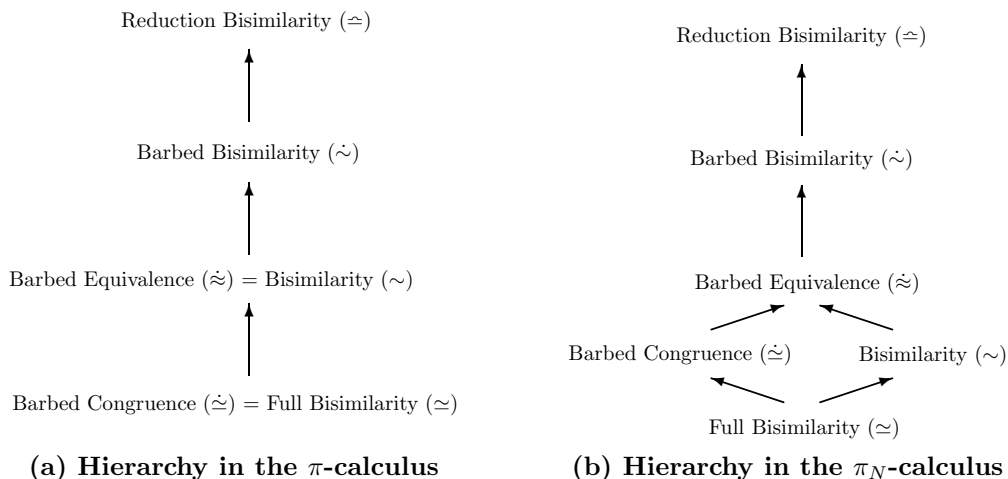


Figure 1: Hierarchy of behavioral equivalences, where an arrow $A \rightarrow B$ expresses that A is strictly included in B

Based upon the early transitional semantics of π_N , we then extend six kinds of behavioral equivalences consisting of reduction bisimilarity, barbed bisimilarity, barbed equivalence, barbed congruence [34], bisimilarity, and full bisimilarity [32, 33] into the π_N -calculus. All these equivalences are defined through certain bisimulations involving internal action and discriminating power. Because of noisy channels, a transition may occur with a certain probability, and thus all the bisimulations are defined quantitatively. Some basic properties of these equivalences have been investigated. Finally, we concentrate on a hierarchy of these behavioral equivalences. In particular, due to the noisy nature of channels, the coincidence of bisimilarity and barbed equivalence which holds in the π -calculus, as well as the coincidence of full bisimilarity and barbed congruence, fails in the π_N -calculus. This hierarchy is shown in Figure 1, together with a corresponding hierarchy in π (see, for example, [43]). Clearly, the hierarchy is helpful to verify the behavioral equivalence of two agents: one can start a proof effort at the middle tier; if this succeeds, one can switch to a finer equivalence; otherwise, one can switch to a coarser equivalence.

The remainder of this paper is structured as follows. We briefly review some basics of the π -calculus in Section 2. After recalling the formal framework of π_N [54] in Section 3.1, we develop the early transitional semantics of the π_N -calculus and present it in two different forms in the remainder of this section. Section 4 is devoted to reduction bisimilarity and barbed bisimilarity, equivalence, and congruence. In the subsequent

section, the other two equivalences, bisimilarity and full bisimilarity, are explored. We complete the hierarchy of these behavioral equivalences in Section 6 and conclude the paper in Section 7.

2 π -calculus

For the convenience of the reader, this section collects some useful facts on the π -calculus from [43]. The syntax, the early transitional semantics, and some notations of the π -calculus are presented in Section 2.1. Section 2.2 briefly reviews several behavioral equivalences of the π -calculus. We refer the reader to [31, 32, 35, 43] and the references therein for an elaborated explanation and the development of the theory of π .

2.1 Basic definitions

We presuppose in the π -calculus a countably infinite set \mathbf{N} of names ranged over by a, b, \dots, x, y, \dots with $\tau \notin \mathbf{N}$, and such names will act as communication channels, variables, and data values. We employ P, Q, R, \dots to serve as meta-variables of agents or process expressions. Processes evolve by performing actions, and the capabilities for action are expressed via the following four kinds of *prefixes*:

$$\pi ::= \bar{x}y \mid x(z) \mid \tau \mid [x = y]\pi,$$

where $\bar{x}y$, an *output prefix*, is capable of sending the name y via the name x ; $x(z)$, an *input prefix*, is capable of receiving any name via x ; τ , the *silent prefix*, is an internal action; and $[x = y]\pi$, a *match prefix*, has the capability π whenever x and y are the same name.

We now recall the syntax of the π -calculus.

Definition 2.1. The *processes* and the *summations* of the π -calculus are given respectively by

$$\begin{aligned} P &::= M \mid P|P' \mid (\nu z)P \mid !P \\ M &::= \mathbf{0} \mid \pi.P \mid M + M'. \end{aligned}$$

In the definition above, $\mathbf{0}$ is a designated process symbol that can do nothing. A *prefix* $\pi.P$ has a single capability expressed by π ; the agent P cannot proceed until that capability has been exercised. A *sum* $P + Q$ represents an agent that can enact either P or Q . A *parallel composition* $P|Q$ represents the combined behavior of P and Q executing in parallel, that is, P and Q can act independently, and may also communicate if one performs an output and the other performs an input along the same channel. The *restriction* operator (νz) in $(\nu z)P$ acts as a static binder for the name z in P . In addition, the input prefix $x(z)$ also binds the name z . The agent $!P$, called *replication*, can be thought of as an infinite composition $P|P|\dots$ or, equivalently, as a process satisfying the equation $!P = P|!P$. Iterative or arbitrarily long behavior can also be described by an alternative mechanism, the so-called recursion. It turns out that replication can encode the recursive definition (see, for example, Section 9.5 in [31]).

Let us introduce some syntactic notations before going forward. As mentioned above, both input prefix and restriction bind names, and we can define the *bound names* $\text{bn}(P)$ as those with a bound occurrence in P and the *free names* $\text{fn}(P)$ as those with a not

bound occurrence. We write $\mathbf{n}(P)$ for the names of P , namely, $\mathbf{n}(P) = \mathbf{fn}(P) \cup \mathbf{bn}(P)$, and sometimes use the abbreviation $\mathbf{fn}(P, Q)$ for $\mathbf{fn}(P) \cup \mathbf{fn}(Q)$.

A *substitution* is a function from names to names that is the identity except on a finite set. We write $\{y/x\}$ for the substitution that maps x to y and is identity for all other names, and in general $\{y_1, \dots, y_n/x_1, \dots, x_n\}$, where the x_i 's are pairwise distinct, for a function that maps each x_i to y_i . We use σ to range over substitutions, and write $x\sigma$, or sometimes $\sigma(x)$, for σ applied to x . The process $P\sigma$ is P where all free names x are replaced by $\sigma(x)$, with changes of some bound names (i.e., alpha-conversion) wherever needed to avoid name captures.

For later need, we fix some notational conventions: A sequence of distinct restrictions $(\nu z_1) \cdots (\nu z_n)P$ is often abbreviated to $(\nu z_1 \cdots z_n)P$, or just $(\nu \tilde{z})P$ when n is not important. We sometimes elide a trailing $\mathbf{0}$, writing α for the process $\alpha.\mathbf{0}$, where this cannot cause confusion. We also follow generally used operator precedence on processes.

For our purpose of investigating barbed equivalence, we only recall the early transition rules here; the reader may refer to [32, 43] for the late one. The transition rules are nothing other than inference rules of labeled transition relations on processes. The transition relations are labeled by the *actions*, of which there are four kinds: the silent action τ , input actions xy , free output actions $\bar{x}y$, and bound output actions $\bar{x}(y)$. The first action is internal action, the second is receiving the name y via the name x , the third is sending y via x , and the last is sending a fresh name via x . Let α, β, \dots range over actions. We write Act for the set of actions. If $\alpha = xy, \bar{x}y$, or $\bar{x}(y)$, then x is called the *subject* and y is called the *object* of α . The *free names* and *bound names* of an action α are given by

$$\mathbf{fn}(\alpha) = \begin{cases} \emptyset, & \text{if } \alpha = \tau \\ \{x, y\}, & \text{if } \alpha = xy \text{ or } \bar{x}y \\ \{x\}, & \text{if } \alpha = \bar{x}(y) \end{cases}$$

and

$$\mathbf{bn}(\alpha) = \begin{cases} \emptyset, & \text{if } \alpha = \tau, xy, \text{ or } \bar{x}y \\ \{y\}, & \text{if } \alpha = \bar{x}(y). \end{cases}$$

The set of names, $\mathbf{n}(\alpha)$, of α is $\mathbf{fn}(\alpha) \cup \mathbf{bn}(\alpha)$.

The transition relation labeled by α will be written as $\xrightarrow{\alpha}$. Thus, $P \xrightarrow{\tau} Q$ will express that P can evolve invisibly to Q ; $P \xrightarrow{xy} Q$ will express that P can receive y via x and become Q ; $P \xrightarrow{\bar{x}y} Q$ will express that P can send y via x and evolve to Q ; and $P \xrightarrow{\bar{x}(y)} Q$ will express that P evolves to Q after sending a fresh name via x .

We are now in the position to review the labeled transition semantics of the π -calculus. The (*early*) *transition relations*, $\{\xrightarrow{\alpha} : \alpha \in Act\}$, are defined by the rules in Table 1. Note that four rules are elided from the table: the symmetric forms Sum-R, Par-R, Comm-R, and Close-R of Sum-L, Par-L, Comm-L, and Close-L, respectively.

We do not discuss and illustrate the rules, and only remark that the side conditions in Par-L, Par-R, Close-L, Close-R, Rep-Close can always be satisfied by changing the object of a bound-output action.

2.2 Behavioral equivalences

In this subsection, we recall the notions of reduction bisimilarity, barbed bisimilarity, barbed equivalence, barbed congruence, bisimilarity, and full bisimilarity of the π -calculus studied in [32, 34, 40, 33, 43]. A hierarchy of them is also recorded.

Out	$\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	Inp	$\frac{}{x(z).P \xrightarrow{xy} P\{y/z\}}$
Tau	$\frac{}{\tau.P \xrightarrow{\tau} P}$	Mat	$\frac{\pi.P \xrightarrow{\alpha} P'}{[x = x]\pi.P \xrightarrow{\alpha} P'}$
Sum-L	$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	Par-L	$\frac{P \xrightarrow{\alpha} P'}{P Q \xrightarrow{\alpha} P' Q} \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$
Comm-L	$\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P Q \xrightarrow{\tau} P' Q'}$	Close-L	$\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xz} Q'}{P Q \xrightarrow{\tau} (\nu z)(P' Q')} \quad z \notin \text{fn}(Q)$
Res	$\frac{P \xrightarrow{\alpha} P'}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \quad z \notin \text{n}(\alpha)$	Open	$\frac{P \xrightarrow{\bar{x}z} P'}{(\nu z)P \xrightarrow{\bar{x}(z)} P'} \quad z \neq x$
Rep-Act	$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'!P}$	Rep-Comm	$\frac{P \xrightarrow{\bar{x}y} P' \quad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} P' P''!P}$
Rep-Close	$\frac{P \xrightarrow{\bar{x}(z)} P' \quad P \xrightarrow{xz} P''}{!P \xrightarrow{\tau} ((\nu z)(P' P''))!P} \quad z \notin \text{fn}(P)$		

Table 1: Early transition rules of the π -calculus

Let us begin with reduction bisimilarity.

Definition 2.2. A relation \mathcal{R} is a *reduction bisimulation* if whenever $(P, Q) \in \mathcal{R}$,

- (1) $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ for some Q' with $(P', Q') \in \mathcal{R}$;
- (2) $Q \xrightarrow{\tau} Q'$ implies $P \xrightarrow{\tau} P'$ for some P' with $(P', Q') \in \mathcal{R}$.

Reduction bisimilarity, denoted \simeq , is the union of all reduction bisimulations.

A somewhat stronger notion than reduction bisimilarity but still very weak is barbed bisimilarity, which takes observability into account. To express observability formally, we need the notion of *observability predicate* $P \downarrow_\theta$, where θ is an arbitrary name or co-name, namely $\theta \in \mathbf{N} \cup \{\bar{a} : a \in \mathbf{N}\}$. We say that $P \downarrow_a$ if P can perform an input action with subject a ; and $P \downarrow_{\bar{a}}$ if P can perform an output action with subject a .

Definition 2.3. A relation \mathcal{R} is a *barbed bisimulation* if whenever $(P, Q) \in \mathcal{R}$,

- (1) $P \downarrow_\theta$ implies $Q \downarrow_\theta$, and vice versa;
- (2) $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ for some Q' with $(P', Q') \in \mathcal{R}$;
- (3) $Q \xrightarrow{\tau} Q'$ implies $P \xrightarrow{\tau} P'$ for some P' with $(P', Q') \in \mathcal{R}$.

Barbed bisimilarity, written $\dot{\sim}$, is the union of all barbed bisimulations.

Based upon barbed bisimulation, we have the following definition.

Definition 2.4. Two processes P and Q are called *barbed equivalent*, denoted $P \dot{\sim} Q$, if $P|R \dot{\sim} Q|R$ for any R .

We are going to introduce barbed congruence, which is stronger than barbed equivalence. To this end, we need an auxiliary notion.

Definition 2.5. *Process contexts* \mathcal{C} are given by the syntax

$$\mathcal{C} ::= [\] \mid \pi.\mathcal{C} + M \mid \mathcal{C}|P \mid P|\mathcal{C} \mid (\nu z)\mathcal{C} \mid !\mathcal{C}.$$

We denote by $\mathcal{C}[P]$ the result of filling the hole $[\]$ in the context \mathcal{C} with the process P . The *elementary contexts* are $\pi.[\] + M$, $[\]|P$, $P|[\]$, $(\nu z)[\]$, and $![\]$.

Now, we can make the following definition.

Definition 2.6. Two processes P and Q are said to be *barbed congruent*, written $P \simeq Q$, if $\mathcal{C}[P] \sim \mathcal{C}[Q]$ for every process context \mathcal{C} .

Let us continue to define bisimilarity and an associated congruence, full bisimilarity.

Definition 2.7. (*Strong*) *bisimilarity* is the largest symmetric relation, \sim , such that whenever $P \sim Q$, $P \xrightarrow{\alpha} P'$ implies $Q \xrightarrow{\alpha} Q'$ for some Q' with $P' \sim Q'$.

Definition 2.8. Two processes P and Q are *full bisimilar*, written $P \simeq Q$, if $P\sigma \sim Q\sigma$ for every substitution σ .

Finally, we summarize a hierarchy of the behavioral equivalences above, which is depicted in Figure 1(a).

Theorem 2.9.

- (1) $\simeq \subseteq \approx \subseteq \sim \subseteq \simeq$; each of the inclusions can be strict.
- (2) $\sim = \approx$ and $\simeq = \simeq$.

For a proof of the above theorem, the reader is referred to Sections 2.1 and 2.2.1 (Lemma 2.2.7 and Theorem 2.2.9) of [43]. We remark that barbed bisimilarity, congruence, and equivalence were introduced in [34], the basic theory of bisimilarity and full bisimilarity was established in [32, 33], and the assertion (2) of Theorem 2.9 was first proved in [40].

3 π -calculus with noisy channels

In the last section, we made an implicit assumption that all communication channels in the π -calculus are noiseless. In the present section, such an assumption is removed and the π -calculus with noisy channels, the π_N -calculus, is explored. The first subsection is devoted to introducing the original formal framework of the π_N -calculus due to Ying [54], which is based on the late transitional semantics of π_N . An early transitional semantics of π_N is proposed in Section 3.2. To cope with transitions of π_N well, we group the transitions according to their sources in Section 3.3.

3.1 Late transitional semantics of π_N

This subsection reviews briefly some basic notions of the π_N -calculus from [54], including noisy channels and the late transitional semantics.

A fundamental assumption in the π_N -calculus [54] is that communication channels may be noisy; that is, their inputs are subject to certain disturbances in transmission. In other words, the communication situation is conceived as that an input is transmitted through a channel and the output is produced at the end of the channel, but the output

is often not completely determined by the input. In Shannon's information theory [45], a mathematical model of channels from statistic communication theory, the noisy nature is usually described by a probability distribution over the output alphabet. This distribution of course depends on the input and in addition it may depend on the internal state of the channel.

A simpler but still very valuable class of noisy channels is memoryless channels, on which the π_N -calculus is based. In such channels, it is assumed that any output does not depend on the internal state of the channel, and moreover, the outputs of any two different inputs are independent. It turns out that a memoryless channel can be completely characterized by its channel matrix

$$[p(y|x)]_{x,y}$$

where $p(y|x)$ is the conditional probability of outputting y when the input is x , and the subscripts x and y run over all inputs and outputs, respectively. Clearly, $p(y|x) \geq 0$, and by definition, we have that $\sum_y p(y|x) = 1$ for any input x .

The syntax of the π_N -calculus is completely the same as that of the π -calculus. The essential difference between π_N and π is that π_N takes the noise of communication channels into consideration, which means that the receiver cannot always get exactly what the sender delivers. As mentioned above, the noisy channels in π_N are assumed to be memoryless, and thus we may suppose that each name $x \in \mathbf{N}$ has a channel matrix

$$M_x = [p_x(z|y)]_{y,z \in \mathbf{N}}$$

where $p_x(z|y)$ is the probability that the receiver will get the name z at the output when the sender emits the name y along the channel x .

For the π -calculus, there are two kinds of transitional semantics. In Section 2.1, the input rule is a transition $x(z).P \xrightarrow{xy} P\{y/z\}$, which expresses that $x(z).P$ can receive the name y via x and evolve to $P\{y/z\}$. An action of the form xy records both the name used for receiving and the name received. The placeholder z is instantiated early, namely when the input by the receiver is inferred. Hence the name "early" semantics. In the literature on π , the first way to treat the semantics for input, the late transitional semantics [32], adopts the input rule $x(z).P \xrightarrow{x(z)} P$, where the label $x(z)$ contains a placeholder z for the name to be received, rather than the name itself. In this context, the input action of the form $x(z)$ which replaces the action xy in the early transitional semantics can be instantiated late, that is, it can be instantiated when a communication is inferred. Therefore, the early and late terminology is based upon when a name (placeholder) is instantiated in inferring an interaction. In fact, there is a very close relationship between the two kinds of semantics (see, for example, Lemma 4.3.2 in [43]) which allows us to freely use the early or late semantics as convenient.

Let us write Act_l for the set of actions in the late transitional semantics, namely, $Act_l = \{\tau, x(y), \bar{x}y, \bar{x}(y) : x, y \in \mathbf{N}\}$. If $\alpha = x(y)$, we set $\text{fn}(\alpha) = \{x\}$ and $\text{bn}(\alpha) = \{y\}$. The structural operational semantics of π_N in [54] is based upon the late transitional semantics and is given by a family of probabilistic transition relations \vdash^α_p ($\alpha \in Act_l, p \in (0, 1]$) displayed in Table 2. The table omits the symmetric forms of SUM-L, PAR-L, COMM-L, and CLOSE-L; note also that the rule IDE for agent identifiers in [54] is replaced by the rules REP-ACT, REP-COMM, and REP-CLOSE since we are using an equivalent notion, replications, instead of agent identifiers in the syntax. The arrow $\vdash \rightarrow$

OUT	$\frac{}{\bar{x}y.P \xrightarrow[p]{\bar{x}z} P} \quad p = p_x(z y) > 0$	INP	$\frac{}{x(z).P \xrightarrow{1}{x(z)} P}$
TAU	$\frac{}{\tau.P \xrightarrow{1}{\tau} P}$	MAT	$\frac{\pi.P \xrightarrow[p]{\alpha} P'}{[x = x]\pi.P \xrightarrow[p]{\alpha} P'}$
SUM-L	$\frac{P \xrightarrow[p]{\alpha} P'}{P + Q \xrightarrow[p]{\alpha} P'}$		
PAR-L	$\frac{P \xrightarrow[p]{\alpha} P'}{P Q \xrightarrow[p]{\alpha} P' Q} \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$		
COMM-L	$\frac{P \xrightarrow[p]{\bar{x}y} P' \quad Q \xrightarrow{1}{x(z)} Q'}{P Q \xrightarrow[p]{\tau} P' Q'\{y/z\}}$	CLOSE-L	$\frac{P \xrightarrow[p]{\bar{x}(z)} P' \quad Q \xrightarrow{1}{x(z)} Q'}{P Q \xrightarrow[p]{\tau} (\nu z)(P' Q')}$
OPEN	$\frac{P \xrightarrow[p]{\bar{x}y} P'}{(\nu y)P \xrightarrow[p]{\bar{x}(y)} P'} \quad y \neq x$		
RES	$\frac{P \xrightarrow[p]{\alpha} P'}{(\nu z)P \xrightarrow[p]{\alpha} (\nu z)P'} \quad z \notin \text{n}(\alpha)$	REP-ACT	$\frac{P \xrightarrow[p]{\alpha} P'}{!P \xrightarrow[p]{\alpha} P' !P}$
REP-COMM	$\frac{P \xrightarrow[p]{\bar{x}y} P' \quad P \xrightarrow{1}{x(z)} P''}{!P \xrightarrow[p]{\tau} P' P''\{y/z\}!P}$	REP-CLOSE	$\frac{P \xrightarrow[p]{\bar{x}(z)} P' \quad P \xrightarrow{1}{x(z)} P''}{!P \xrightarrow[p]{\tau} (\nu z)(P' P'')!P}$

Table 2: Late transition rules of π_N

α	kind	$\text{barb}(\alpha)$	$\text{subj}(\alpha)$	$\text{obj}(\alpha)$	$\text{n}(\alpha)$	$\alpha\sigma$
τ	Silent	τ	—	—	\emptyset	τ
xy	Input	x	x	y	$\{x, y\}$	$x\sigma y\sigma$
$\bar{x}y$	Noisy free output	\bar{x}	x	y	$\{x, y\}$	$\bar{x}\sigma y\sigma$
$\bar{x}(y)$	Noisy bound output	\bar{x}	x	y	$\{x, y\}$	$\bar{x}\sigma(y\sigma)$

Table 3: Terminology and notation for actions

is used to distinguish the late relations from the early, and the probability values p arise entirely from the noise of communication channels. The OUT rule, which represents the noisy nature of channels, is the unique one that all differences between π_N and π come from. It means that the process $\bar{x}y.P$ of output prefix form sends the name y via the channel x , but what the receiver gets at the output of this channel may not be y due to noise residing in it, and a name z will be received with the probability $p_x(z|y)$.

3.2 Early transitional semantics of π_N

For the π -calculus, it turns out that the early transitional semantics is somewhat simpler than the late one for investigating behavioral equivalences. The reason is that the input action in the early semantics is instantiated early and thus we need not check all possible instantiations of a placeholder. In light of this, we pay our attention to the early transitional semantics of π_N in this subsection. This semantics is not, however, a direct translation of Table 2, as we will see shortly.

Out	$\frac{}{\overline{xy}.P \xrightarrow[p]{\overline{xz}} P} \quad p = p_x(z y) > 0$	Inf	$\frac{}{x(z).P \xrightarrow{xy}_1 P\{y/z\}}$
Tau	$\frac{}{\tau.P \xrightarrow{1} P}$	Mat	$\frac{\pi.P \xrightarrow{\alpha}_p P'}{[x=x]\pi.P \xrightarrow{\alpha}_p P'}$
Sum-L	$\frac{P \xrightarrow{\alpha}_p P'}{P+Q \xrightarrow{\alpha}_p P'}$	Par-L	$\frac{P \xrightarrow{\alpha}_p P'}{P Q \xrightarrow{\alpha}_p P' Q}$
Comm-L	$\frac{P \xrightarrow{\overline{xy}}_p P' \quad Q \xrightarrow{xy}_1 Q'}{P Q \xrightarrow{\tau}_p P' Q'}$	Close-L	$\frac{P \xrightarrow{\overline{x}(y)}_p P' \quad Q \xrightarrow{xy}_1 Q'}{P Q \xrightarrow{\tau}_p (\nu y)(P' Q')}$
Res	$\frac{P \xrightarrow{\alpha}_p P'}{(\nu z)P \xrightarrow{\alpha}_p (\nu z)P'} \quad z \notin n(\alpha)$	Open-Out	$\frac{P \xrightarrow{\overline{xy}}_p P'}{(\nu y)P \xrightarrow{\overline{x}(y)}_p P'} \quad y \neq x$
Open-Inf	$\frac{P \xrightarrow{xy}_1 P'}{(\nu y)P \xrightarrow{xy}_1 P'} \quad y \neq x$	Rep-Act	$\frac{P \xrightarrow{\alpha}_p P'}{!P \xrightarrow{\alpha}_p P' !P}$
Rep-Comm	$\frac{P \xrightarrow{\overline{xy}}_p P' \quad P \xrightarrow{xy}_1 P''}{!P \xrightarrow{\tau}_p P' P'' !P}$	Rep-Close	$\frac{P \xrightarrow{\overline{x}(y)}_p P' \quad P \xrightarrow{xy}_1 P''}{!P \xrightarrow{\tau}_p (\nu y)(P' P'') !P}$

Table 4: Early transition rules of π_N

Like the late transitional semantics of π_N , the early semantics of π_N is also given in terms of probabilistic transition relations. A probabilistic transition in the π_N is of the form

$$P \xrightarrow[p]{\alpha} Q$$

where P and Q are two processes, $\alpha \in Act = \{\tau, xy, \overline{xy}, \overline{x}(y) : x, y \in \mathbf{N}\}$, and $p \in (0, 1]$. The intuitive meaning of this transition is that the agent P performs action α and becomes Q , with probability p . It should be pointed out that although the actions here are the same as those in the early semantics of π , the meanings of them are not completely identical. More concretely, τ and xy still represent an internal action and an input of a name y alone channel x , respectively, but \overline{xy} represents output of a name y via a noisy channel x that changes the intended output of some name into y with a certain probability, and $\overline{x}(y)$ represents output of a bound name y via a noisy channel x that changes the intended output of some name into y with a certain probability. Table 3 displays terminology and notation pertaining to the actions. Its columns list, respectively, the *kind* of an action α , the *barb* of α , the *subject* of α , the *object* of α , the set of *names* of α , and the effect of applying a substitution to α ; some issues different from those of π such as $\overline{x}(y)\sigma$ will be explained subsequently.

The early transition rules of π_N is present in Table 4. As before, we omit the symmetric forms of Sum-L, Par-L, Comm-L, and Close-L. Let us make a brief discussion about the rationale behind the design:

1) Since we follow the assumption of the noisy channels in [54], the Out rule is the same as OUT in Table 2. It shows that the action performed by $\overline{xy}.P$ is not \overline{xy} but \overline{xz} , and the probability $p_x(z|y)$ that y becomes z in channel x is indicated. This is thought of as that noise happens at the end of sending, not at the end of receiving.

2) All rules except for Out and Open-Inf are just simple imitations of the corre-

sponding rules in the π -calculus. Nevertheless, there are two differences: one is that a probability parameter p is taken into account, which is necessary for encoding the noise of channels; the other is that the side conditions in Par-L, Close-L, and Rep-Close are elided. The latter arises entirely from that the condition $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$ is not required when considering the left parallel composition $P|Q$.

The reason for removing the condition $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$ arises from the following consideration. Recall that in π_N [54] it was supposed that free names and bound names are distinct. This assumption inconveniences the use of some transition rules such as Open-Out in our context. Recall also that in π the side condition $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$ of inferring $P|Q$ can be easily satisfied by utilizing alpha-conversion on P . However, this conversion must involve the congruent equation $(\nu z)P \equiv (\nu w)P\{w/z\}$, where $w \notin \text{n}(P)$. It is unfortunate that such a well known and widely used equation in the concurrency community seems to be impracticable for π_N . To see this, let us examine a specific example. Suppose that

$$p_x(y|y) = p_0, \quad p_x(a|y) = p_1, \quad p_x(b|y) = p_2,$$

where $p_0 + p_1 + p_2 = 1$. We choose $P \stackrel{\text{def}}{=} (\nu a)\bar{x}y|x(w).\bar{w}z$ and $Q \stackrel{\text{def}}{=} (\nu b)\bar{x}y|x(w).\bar{w}z$. Assume that $(\nu z)P \equiv (\nu w)P\{w/z\}$ with $w \notin \text{n}(P)$ holds in π_N . Then it is clear that $P \equiv Q$, and thus we can identify P with Q . By the early transition rules of π_N , we get without breaking the side condition $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$ that

$$\begin{aligned} P &\xrightarrow{p_0} \bar{y}z, \quad P \xrightarrow{p_1} (\nu a)\bar{a}z, \quad P \xrightarrow{p_2} \bar{b}z; \\ Q &\xrightarrow{p_0} \bar{y}z, \quad Q \xrightarrow{p_1} \bar{a}z, \quad Q \xrightarrow{p_2} (\nu b)\bar{b}z. \end{aligned}$$

Because $(\nu a)\bar{a}z$ and $(\nu b)\bar{b}z$ are inactive, while $\bar{x}z$ and $\bar{b}z$ are capable of sending z , this forces that $p_1 = p_2 = 0$, and thus $p_0 = 1$. It means that the channel x is noiseless when outputting y ; this is absurd because x and y can be taken arbitrarily, including noisy channels.

In light of the previous discussion, it seems better to do away with the congruent equation $(\nu z)P \equiv (\nu w)P\{w/z\}$, $w \notin \text{n}(P)$. As a result, we cannot keep the side condition $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$, because otherwise the associative law among process interactions would be violated. To see this, one may consider the processes $x(z)|((\nu y)\bar{x}y|\bar{y}w)$ and $(x(z)|(\nu y)\bar{x}y)|\bar{y}w$. Since the associativity is a very important property of mobile systems that the π -calculus has intended to understand, we would not like to destroy it. As a consequence, the side conditions in Par-L, Close-L, and Rep-Close are removed, and this yields that the proposed early transitional semantics of π_N is somewhat different from that of π when considering only noiseless channels.

Recall also that in the π -calculus, a private name in a process P is local, meaning it can be used only for communication between components within P . Such a private name cannot immediately be used as a port for communication between P and its environment; in fact, because P may rename its private names, these names are not known by the environment. Note, however, that not allowing the congruence $(\nu z)P \equiv (\nu w)P\{w/z\}$, together with the noise of channels, makes the private names in π_N somewhat public. More concretely, since a process P in π_N cannot rename its private names, these names may appear in the environment of P . In addition, every name may be confused with any other name because of the noise of channels. Nevertheless, private names are needed in π_N since a private name, say z , can at least be used to preclude a process from communicating with its environment via the port z .

3) An Open-Inp rule is added. This arises from two aspects of consideration: One is that if a channel is capable of inputting, then it should have the ability of arbitrary inputting. In other words, if $P \xrightarrow{xy}_1 Q$, then restricting y to P should not prevent the channel x from inputting y . The other aspect is that if an agent can receive a name from outside, then the name may be thought of as open and the scope of the restriction may be extended. Technically, the Open-Inp rule derives from the invalidation of the congruent equation $(\nu z)P \equiv (\nu w)P\{w/z\}$, $w \notin \text{n}(P)$, because without the equation, bound names cannot be changed and the free name in an input may clash with bound names. We remark that such a rule is not necessary in the π -calculus, since all the bound names in π can be renamed by alpha-conversion.

Let us continue introducing some notions. As in π , the input prefix $x(z)$ and the restriction (νz) bind the name z . In view of 2) above, in the π_N -calculus we need to differentiate between the bound names in $(\nu z).Q$ and $x(z).Q$. A bound name z is called *strongly bound* in P if it lies within some sub-term $x(z).Q$ of P . As shown in Definition 3.3 (1), we permit of changing a strongly bound name into a fresh name, which is called *strong alpha-conversion*. Any bound name that is not strongly bound is said to be *weakly bound*. An occurrence of a name in a process of π_N is *free* if it is not bound. For instance, in $P \stackrel{\text{def}}{=} (\nu s)x(z).(\nu z)\bar{y}z.\bar{x}s$, the name s is weakly bound, z is strongly bound, and x and y are free. We denote the free names, bound names, strongly bound names, and weakly bound names in a process P by $\text{fn}(P)$, $\text{bn}(P)$, $\text{sbn}(P)$, and $\text{wbn}(P)$, respectively.

We now define the effect of applying a substitution σ to a process P in π_N . Since weakly bound names cannot be converted, the process $P\sigma$ is P where all free names and weakly bound names x are replaced by $x\sigma$, with strong alpha-conversion wherever needed to avoid captures. This means that strongly bound names are changed such that whenever x is replaced by $x\sigma$ then the so obtained occurrence of $x\sigma$ is not strongly bound. For instance,

$$(a(x).(\nu b)\bar{x}b.\bar{c}y.\mathbf{0})\{x, c/y, b\} = a(z).(\nu c)\bar{z}c.\bar{c}x.\mathbf{0}.$$

Clearly, according to the above definition of substitution, we have the following fact.

Lemma 3.1. For any substitution σ ,

- (1) $\mathbf{0}\sigma = \mathbf{0}$;
- (2) $(\pi.P)\sigma = \pi\sigma.P\sigma$;
- (3) $(P + Q)\sigma = P\sigma + Q\sigma$;
- (4) $(P|Q)\sigma = P\sigma|Q\sigma$;
- (5) $((\nu z)P)\sigma = (\nu z\sigma)P\sigma$;
- (6) $(!P)\sigma = !P\sigma$.

Since the notion of free names is not sufficient for defining structural congruence, we introduce an extended notion of free names. Motivated by a similar notion in [54], we define the set of noisy free names to be the set of all free names in a process and those names produced by noise when sending free names. Formally, we have the following.

Definition 3.2. The set of *noisy free names*, denoted $\text{fn}^*(P)$, is defined inductively as follows:

- (1) $\text{fn}^*(\mathbf{0}) = \emptyset$;
- (2) $\text{fn}^*(\bar{x}y.P) = \{x\} \cup \{z \in \mathbf{N} : p_x(z|y) > 0\} \cup \text{fn}^*(P)$;
- (3) $\text{fn}^*(x(z).P) = \{x\} \cup \bigcup_{y \in \mathbf{N}} (\text{fn}^*(P\{y/z\}) \setminus \{y\})$;
- (4) $\text{fn}^*(\tau.P) = \text{fn}^*(P)$;
- (5) $\text{fn}^*([x = y]\pi.P) = \{x, y\} \cup \text{fn}^*(\pi.P)$;
- (6) $\text{fn}^*(P + P') = \text{fn}^*(P|P') = \text{fn}^*(P) \cup \text{fn}^*(P')$;
- (7) $\text{fn}^*((\nu z)P) = \text{fn}^*(P) \setminus \{z\}$;
- (8) $\text{fn}^*(!P) = \text{fn}^*(P)$.

Although $\text{fn}^*(P)$ is an extension of $\text{fn}(P)$, it is not necessarily that $\text{fn}^*(P) \supseteq \text{fn}(P)$. For example, assume that $p_x(z|y) = 1$. Then we see that $\text{fn}^*(\bar{x}y.\mathbf{0}) = \{x, z\}$, while $\text{fn}(\bar{x}y.\mathbf{0}) = \{x, y\}$. If it is required that $p_x(y|y) > 0$ for all $x, y \in \mathbf{N}$, then we indeed have that $\text{fn}^*(P) \supseteq \text{fn}(P)$.

We can now define structural congruence as follows.

Definition 3.3. Two process expressions P and Q in the π_N -calculus are *structurally congruent*, denoted $P \equiv Q$, if we can transform one into the other by using the following equations (in either direction):

- (1) $x(z).P \equiv x(w).P\{w/z\}$ if w is fresh in P .
- (2) Reordering of terms in a summation.
- (3) $M + \mathbf{0} \equiv M$, $P|\mathbf{0} \equiv P$, $P|Q \equiv Q|P$, and $P|(Q|R) \equiv (P|Q)|R$.
- (4) $(\nu z)(P|Q) \equiv P|(\nu z)Q$ if $z \notin \text{fn}^*(P)$, $(\nu z)\mathbf{0} \equiv \mathbf{0}$, and $(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P$.
- (5) $[x = x]\pi.P \equiv \pi.P$.
- (6) $[x = y]\pi.P \equiv \mathbf{0}$ if x and y are distinct.
- (7) $!P \equiv P|!P$.

The noisy free names of two structurally congruent processes are clearly related by the following fact.

Lemma 3.4. If $P \equiv Q$ can be inferred without using (5) and (6) in Definition 3.3, then $\text{fn}^*(P) = \text{fn}^*(Q)$.

We also make an observation on the noisy free names of processes appearing in the same transition.

Lemma 3.5. Let $P \xrightarrow{\alpha}_P P'$ be a transition in π_N .

- (1) If $\alpha = \bar{x}y$, then $x, y \in \text{fn}^*(P)$ and $\text{fn}^*(P') \subseteq \text{fn}^*(P)$.

- (2) If $\alpha = xy$, then $x \in \text{fn}^*(P)$ and $\text{fn}^*(P') \subseteq \text{fn}^*(P, y)$.
- (3) If $\alpha = \bar{x}(y)$, then $x \in \text{fn}^*(P)$ and $\text{fn}^*(P') \subseteq \text{fn}^*(P, y)$.
- (4) If $\alpha = \tau$, then $\text{fn}^*(P') \subseteq \text{fn}^*(P)$.

Proof. The proof is carried out by induction on the depth of inference $P \xrightarrow{\alpha}_p P'$. We need to consider all kinds of transition rules that are possible as the last rule in deriving $P \xrightarrow{\alpha}_p P'$. The assertions (1) and (2) can be verified directly, the proof of (3) needs (1), and the proof of (4) needs the first three. This is a long but routine case analysis, so the details are omitted. \square

We end this subsection by providing some image-finiteness properties of probabilistic transitions. To this end, we suppose that outputting a name can only gives rise to finite noisy names, that is, we adopt the following convention:

Convention 3.6. For any $x, y \in \mathbf{N}$, the set $\{y_i : p_x(y_i|y) > 0\}$ is finite.

The following facts about input and output actions are the corresponding results of Lemmas 1.4.4 and 1.4.5 in [43].

Lemma 3.7.

- (1) If $P \xrightarrow{xy}_1 P'$ and $y \notin \text{fn}(P) \cup \text{wbn}(P)$, then $P \xrightarrow{xz}_1 P'\{z/y\}$ for any z .
- (2) If $P \xrightarrow{xy}_1 P'$ and $z \notin \text{fn}(P) \cup \text{wbn}(P)$, then there is P'' such that $P \xrightarrow{xz}_1 P''$ and $P''\{y/z\} = P'$.
- (3) For any P and x , there exist P_1, \dots, P_n and $y \notin \text{fn}(P) \cup \text{wbn}(P)$ such that if $P \xrightarrow{xz}_1 P'$ then $P' = P_i\{z/y\}$ for some P_i .
- (4) For any P , there are only finitely many x such that $P \xrightarrow{xy}_1 P'$ for some y and P' .

Proof. The first two assertions can be easily proved by induction on inference, and the last two by induction on P . \square

Similar to the above, we have a result about output actions.

Lemma 3.8.

- (1) For any P , there are only finitely many quadruples x, y, p, P' such that $P \xrightarrow{\bar{x}y}_p P'$.
- (2) For any P , there are only finitely many quadruples x, y, p, P' such that $P \xrightarrow{\bar{x}(y)}_p P'$.

Proof. All the two assertions are proved by induction on P . The proof of (1) needs Convention 3.6, and the proof of (2) uses the fact that weakly bound names cannot be converted. \square

3.3 Transition groups of π_N

In the last subsection, the primitive transition rules of π_N have been established. A closer examination, however, shows that at least two confusions may arise from the presentation of these rules. To avoid this, we introduce another presentation, transition groups, in this subsection.

To illustrate our motivation, let us consider a simple example: Take $P \stackrel{\text{def}}{=} \bar{x}y + \bar{x}z$, and suppose that

$$\begin{aligned} p_x(y|y) &= 0.7, \quad p_x(z|y) = 0.1, \quad p_x(s|y) = 0.1, \quad p_x(t|y) = 0.1; \\ p_x(y|z) &= 0.5, \quad p_x(z|z) = 0.3, \quad p_x(s|z) = 0.1, \quad p_x(w|z) = 0.1. \end{aligned}$$

By the transition rules of π_N , we see that

$$P \xrightarrow{\bar{x}y}_{0.7} \mathbf{0}, \quad P \xrightarrow{\bar{x}y}_{0.5} \mathbf{0}, \quad P \xrightarrow{\bar{x}s}_{0.1} \mathbf{0}, \quad \text{and} \quad P \xrightarrow{\bar{x}t}_{0.1} \mathbf{0}.$$

At this point, two confusions arise: One is that there are two probabilistic transitions $P \xrightarrow{\bar{x}y}_{0.7} \mathbf{0}$ and $P \xrightarrow{\bar{x}y}_{0.5} \mathbf{0}$ with the same source and target processes and the same action, but different probability values. In [54], it is thought that the probability of transition $P \xrightarrow{\bar{x}y} \mathbf{0}$ is either 0.7 or 0.5, but which of them is not exactly known and the choice between 0.7 and 0.5 is made by the environment. This understanding that comes from the idea of imprecise probability studied widely by the communities of Statistics and Artificial Intelligence (see, for example, [49]) is very natural. The limit is that more uncertainties are involved in inference. Another way to deal with this problem in the literature on probabilistic processes is to modify probabilistic transition systems. This is done by adding up all possible values of transition probability with the same source and target agents and the same action, and then normalizing them if necessary (for instance, see [47, 26, 48, 51]). Once again, this kind of modification highly complicates the theory of probabilistic processes.

The other confusion is that the information on the origins of some probabilistic transitions is lost. As for the above example, the information that $P \xrightarrow{\bar{x}t}_{0.1} \mathbf{0}$ derives from the first sub-term of P is lost. In addition, we cannot determine from which the transition $P \xrightarrow{\bar{x}s}_{0.1} \mathbf{0}$ arises.

There is a convenient presentation for alleviating the confusions. In fact, we may group all transitions having the same origin. Regarding the example above, we may say that P has two transition groups

$$P\{\xrightarrow{\bar{x}y}_{0.7}\mathbf{0}, \xrightarrow{\bar{x}z}_{0.1}\mathbf{0}, \xrightarrow{\bar{x}s}_{0.1}\mathbf{0}, \xrightarrow{\bar{x}t}_{0.1}\mathbf{0}\} \quad \text{and} \quad P\{\xrightarrow{\bar{x}y}_{0.5}\mathbf{0}, \xrightarrow{\bar{x}z}_{0.3}\mathbf{0}, \xrightarrow{\bar{x}s}_{0.1}\mathbf{0}, \xrightarrow{\bar{x}w}_{0.1}\mathbf{0}\}.$$

Notice that the agent $P \stackrel{\text{def}}{=} \bar{x}y + \bar{x}z$ has a nondeterministic choice between $\bar{x}y$ and $\bar{x}z$, and it is usually thought that such a choice is made by the environment, so we may think that the choice between the transition groups is also made by the environment. In this way, the first confusion is completely excluded, and if we have observed an output $\bar{x}t$ and the environment does not change her choice, then the process has a smaller probability of outputting $\bar{x}z$ and the output $\bar{x}s$ arises necessarily from the first sub-term of P .

In general, we use

$$P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I}$$

to represent a group of probabilistic transitions $P \xrightarrow{\alpha_i}_{p_i} Q_i$, $i \in I$, satisfying $\sum_{i \in I} p_i = 1$; $P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I}$ is called a *transition group*. We omit the indexing set I whenever I is a singleton. In fact, some representations analogous to the transition group have already been used in the literature (see, for example, [44, 20, 46, 15]).

To manipulate transition groups, we need the operator \uplus defined as follows:

$$\{\xrightarrow{p_i} Q_i\}_{i \in I} \uplus \{\xrightarrow{\beta} Q\} = \begin{cases} \{\xrightarrow{p_i+p} Q_i\}_{i \in I}, & \text{if } Q = Q_i \text{ and } \beta = \alpha_i \text{ for some } i \\ \{\xrightarrow{p_i} Q_i\}_{i \in I} \cup \{\xrightarrow{\beta} Q\}, & \text{otherwise;} \end{cases}$$

$$\{\xrightarrow{p_i} Q_i\}_{i \in I} \uplus \{\xrightarrow{p_j} Q_j\}_{j \in J} = (\{\xrightarrow{p_i} Q_i\}_{i \in I} \uplus \{\xrightarrow{p_j} Q_j\}) \uplus \{\xrightarrow{p_k} Q_k\}_{k \in J \setminus \{j\}}.$$

Moreover, if in the same transition group, there are $P \xrightarrow{\alpha_i}_{p_i} Q_i$ and $P \xrightarrow{\alpha_j}_{p_j} Q_j$ with $\alpha_i = \alpha_j$ and $Q_i = Q_j$, then we sometimes combine them into a single one $P \xrightarrow{\alpha_i}_{p_i+p_j} Q_i$ and delete j from the indexing set I . For instance, the transition groups $P\{\xrightarrow{\bar{x}y}_{0.4} Q\} \uplus \{\xrightarrow{\bar{x}y}_{0.6} Q\}$, $P\{\xrightarrow{\bar{x}y}_{0.4} Q, \xrightarrow{\bar{x}y}_{0.6} Q\}$, and $P\{\xrightarrow{\bar{x}y}_1 Q\}$ mean the same thing.

The following result is helpful to group the transitions of π_N .

Lemma 3.9. Let $P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I}$ be a transition group derived from the early transition rules of π_N in Table 4. Then all α_i , $i \in I$, have the same barb.

Proof. It is obvious by the inference rules in the transitional semantics of π_N . \square

In light of Lemma 3.9, we can say that a transition group has a barb, which is defined as the common barb arising from the actions of probabilistic transitions in the transition group. A transition group is called an *output transition group* (respectively, *input transition group*) if its barb is of the form \bar{x} (respectively, x).

Using Lemma 3.9, the transition rules in Table 4 are grouped in Table 5.

For later need, we introduce one more notation. A function μ from Ω to the closed unit interval $[0, 1]$ is called a *probability distribution* on Ω if $\sum_{x \in \Omega} \mu(x) = 1$. By $\mathcal{D}(\Omega)$ we denote the set of all probability distributions on the set Ω . If $\mu \in \mathcal{D}(\Omega \times \Gamma)$, $\omega \in \Omega$, and $S \subseteq \Gamma$, we define $\mu(\omega, S) = \sum_{s \in S} \mu(\omega, s)$.

Observe that each transition group, say $P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I}$, gives rise to a probabilistic distribution μ on $Act \times Proc$, where μ is defined by

$$\mu(\alpha, Q) = \begin{cases} p, & \text{if } P \xrightarrow{\alpha}_p Q \text{ belongs to } P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we sometimes write $P \longrightarrow \mu$ or $P \xrightarrow{\text{barb}(\alpha_i)} \mu$ for $P\{\xrightarrow{\alpha_i}_{p_i} Q_i\}_{i \in I}$, where $\text{barb}(\alpha_i)$ is the barb of the transition group.

We end this subsection with two properties of transition groups. The first one corresponds to the image-finiteness of transition relations in π . Here, by image-finiteness we mean that for any process P and action α in π , there are only finitely many processes Q such that $P \xrightarrow{\alpha} Q$.

Lemma 3.10. Keep Convention 3.6. Then for every $\alpha \in Act$ and $P \in Proc$, the set

$$\{\mu|_{\{\alpha\} \times Proc} : P \longrightarrow \mu\}$$

is finite, where $\mu|_{\{\alpha\} \times Proc}$ is the restriction of μ to $\{\alpha\} \times Proc$.

Proof. It follows immediately from Lemmas 3.7 and 3.8. \square

The other property of transition groups is concerned with applying substitution to transitions. To state it, we need the next definition.

$\text{Out} \frac{}{\overline{xy}.P\{\overline{xy}_i \rightarrow_{p_i} P\}_{i \in I}} \quad I = \{i : p_i = p_x(y_i y) > 0\}$	$\text{Tau} \frac{}{\tau.P\{\rightarrow_1 P\}}$
$\text{Inp} \frac{}{x(z).P\{\xrightarrow{xy}_1 P\{y/z\}\}}$	$\text{Mat} \frac{\pi.P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}{[x = x]\pi.P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}$
$\text{Sum-L} \frac{P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}{P + Q\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}$	$\text{Par-L} \frac{P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}{P Q\{\xrightarrow{\alpha_i}_{p_i} P_i Q\}_{i \in I}}$
$\text{Comm-L} \frac{P\{\overline{xy}_i \rightarrow_{p_i} P_i\}_{i \in I'} \uplus \{\overline{x}(y_i) \rightarrow_{p_i} P_i\}_{i \in I''} \quad Q\{\xrightarrow{xy}_1 Q_i\}}{P Q\{\xrightarrow{\tau}_{p_i} P_i Q_i\}_{i \in I'} \uplus \{\xrightarrow{\tau}_{p_i} (\nu y_i)(P_i Q_i)\}_{i \in I''}}$	
$\text{Res-Out} \frac{P\{\overline{xy}_i \rightarrow_{p_i} P_i\}_{i \in I'} \uplus \{\overline{x}(y_i) \rightarrow_{p_i} P_i\}_{i \in I''}}{(\nu y_j)P\{\overline{xy}_i \rightarrow_{p_i} (\nu y_j)P_i\}_{i \in I' \setminus \{j\}} \uplus \{\overline{x}(y_i) \rightarrow_{p_i} (\nu y_j)P_i\}_{i \in I'' \setminus \{j\}} \uplus \{\overline{x}(y_j) \rightarrow_{p_j} P_j\}} \quad y_j \neq x$	
$\text{Res-Inp} \frac{P\{\xrightarrow{xy}_1 P'\}}{(\nu z)P\{\xrightarrow{xy}_1 (\nu z)P'\}} \quad z \neq x, y$	$\text{Res-Tau} \frac{P\{\xrightarrow{\tau}_{p_i} P_i\}_{i \in I}}{(\nu z)P\{\xrightarrow{\tau}_{p_i} (\nu z)P_i\}_{i \in I}}$
$\text{Open-Inp} \frac{P\{\xrightarrow{xy}_1 P'\}}{(\nu y)P\{\xrightarrow{xy}_1 P'\}} \quad y \neq x$	$\text{Rep-Act} \frac{P\{\xrightarrow{\alpha_i}_{p_i} P_i\}_{i \in I}}{!P\{\xrightarrow{\alpha_i}_{p_i} P_i !P\}_{i \in I}}$
$\text{Rep-Comm} \frac{P\{\overline{xy}_i \rightarrow_{p_i} P_i\}_{i \in I'} \uplus \{\overline{x}(y_i) \rightarrow_{p_i} P_i\}_{i \in I''} \quad P\{\xrightarrow{xy}_1 P'_i\}}{!P\{\xrightarrow{\tau}_{p_i} P_i P'_i !P\}_{i \in I'} \uplus \{\xrightarrow{\tau}_{p_i} (\nu y_i)(P_i P'_i) !P\}_{i \in I''}}$	

Table 5: Transition group rules of π_N

Definition 3.11.

(1) A substitution σ is said to be *consistent* with weakly bound names of a process P if $z\sigma \in \text{wbn}(P\sigma)$ implies $z \notin \text{fn}^*(P)$.

(2) A substitution σ is said to be *compatible* with a channel $x \in \mathbf{N}$ if for any $y, z \in \mathbf{N}$ it holds that

$$p_{x\sigma}(u|y\sigma) = \sum_{z\sigma=u} p_x(z|y).$$

We remark that the definition of compatibility here is a special case of Definition 1 in [54]. A transition group under a consistent and compatible substitution has the following property.

Proposition 3.12. Let $P\{\xrightarrow{p_i} P_i\}_{i \in I}$ be a transition group. Suppose that σ is a substitution satisfying the following conditions:

- 1) σ is consistent with weakly bound names of all processes appearing in the inference for deriving the transition group;
- 2) σ is compatible with the subjects of output actions appearing in the inference for deriving the transition group.

Then, there is a transition group $P\sigma\{\xrightarrow{p_i} P_i\sigma\}_{i \in I}$.

Proof. It follows from a case by case check of all possible transition groups $P\{\xrightarrow{p_i} P_i\}_{i \in I}$. The condition 1) is used to compute probability of $P\sigma \xrightarrow{p_i} P_i\sigma$, and the condition 2) is required when restricting P . We only consider the case of Out and omit the remainder. In the case of Out, we may assume that $P = \overline{x}y.R$, $p_x(y_i|y) = p_i$, $\alpha_i = \overline{x}y_i$, and $P_i = R$. Then the transition group is $\overline{x}y.R\{\xrightarrow{p_i} R\}_{i \in I}$. This gives a transition group $\overline{x}\sigma y\sigma.R\sigma\{\xrightarrow{q_j} R\sigma\}_{j \in J}$ by the definition of substitution, where $J = \{j : q_j = p_{x\sigma}(u_j|y\sigma) > 0\}$. It follows from the condition 2) that

$$q_j = p_{x\sigma}(u_j|y\sigma) = \sum_{y_i\sigma=u_j} p_x(y_i|y) = \sum_{i \in I_j} p_x(y_i|y) = \sum_{i \in I_j} p_i,$$

where $I_j = \{i \in I : y_i\sigma = u_j\}$. Therefore,

$$\begin{aligned} P\sigma\{\xrightarrow{p_i} P_i\sigma\}_{i \in I} &= \overline{x}\sigma y\sigma.R\sigma\{\xrightarrow{p_i} R\sigma\}_{i \in I} \\ &= \overline{x}\sigma y\sigma.R\sigma\{\xrightarrow{\sum_{i \in I_j} p_i} R\sigma\}_{j \in J} \\ &= \overline{x}\sigma y\sigma.R\sigma\{\xrightarrow{q_j} R\sigma\}_{j \in J}, \end{aligned}$$

as desired. \square

4 Barbed equivalence

Having built the transition group rules, we can now turn to several behavioral equivalences in π_N . As mentioned in the last section, collecting the probabilistic transitions arising from a noisy channel into a group can alleviate some confusions, so the transition group provides a useful way of defining behavioral equivalences. Of course, other

ways which do not use the transition group are possible (for example, Definition 4 in [54]). Recall that in π_N two kinds of actions, noisy free output $\bar{x}y$ and noisy bound output $\bar{x}(y)$, have the same barb x . Clearly, not all observers can detect whether an emitted name is bound. It is therefore natural to ask what is the effect of reducing this discriminatory power. In this section, the concepts about behavioral equivalences are based upon the assumption that the observer is limited to seeing whether an action is enabled on a given channel; behavioral equivalences with a more powerful observer will be studied in the next section. Because the behavior of a process in π_N is evidently dependent on the noise probability distribution, it is better to parameterize a behavioral equivalence with the noise. However, for simplicity we assume that all processes in a definition of behavioral equivalence are considered under the same noisy environment, that is, the channel matrix of every name is fixed.

Let us start with some basic notions. The *transitive closure* of a binary relation \mathcal{R} on *Proc* is the minimal transitive relation \mathcal{R}^* on *Proc* that contains \mathcal{R} , that is, if $(P, Q) \in \mathcal{R}^*$, then there exist $P_0, \dots, P_n \in \text{Proc}$ satisfying that $P = P_0$, $Q = P_n$, and $(P_{i-1}, P_i) \in \mathcal{R}$ for $i = 1, \dots, n$. Thus, if \mathcal{R} and \mathcal{S} are two equivalence relations on *Proc*, then so is $(\mathcal{R} \cup \mathcal{S})^*$. It turns out that $(\mathcal{R} \cup \mathcal{S})^*$ is the smallest equivalence relation containing both \mathcal{R} and \mathcal{S} . For any equivalence relation \mathcal{R} on *Proc*, we denote by Proc/\mathcal{R} the set of equivalence classes induced by \mathcal{R} .

The definition below only takes the internal action into account.

Definition 4.1. An equivalence relation \mathcal{R} on *Proc* is a *reduction bisimulation* if whenever $(P, Q) \in \mathcal{R}$, $P \xrightarrow{\tau} \mu$ implies $Q \xrightarrow{\tau} \eta$ for some η satisfying $\mu(\tau, C) = \eta(\tau, C)$ for any $C \in \text{Proc}/\mathcal{R}$.

Note that the above reduction bisimulation and also other subsequent notions on behavioral equivalences are defined in the same style as Larsen-Skou's probabilistic bisimulation [26], that is, the probabilities of reaching an equivalence class have to be computed. Recall that in a non-probabilistic setting we simply require that $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ and $(P', Q') \in \mathcal{R}$, that is, it is enough that the agent Q has a possibility to imitate the step of the agent P . However, if we work in a probabilistic setting, in addition to need that the second agent is able to imitate the first one, it is also reasonable to require that he does it with the same probability. In other words, we have to consider all the possible ways to imitate the execution of the action, and thus we have to add the probabilities associated with these possibilities. Correspondingly, summing up the probabilities of reaching an equivalence class replaces the condition $(P', Q') \in \mathcal{R}$ in a non-probabilistic setting. It should be noted that the treatment of probabilities here is slightly different from that of λ -bisimulation in [54], where a higher probability of an action is allowed to simulate the same action. In the literature, lumping equivalence, a notion on Markov chains to aggregate state spaces [14, 21], was also defined by summing up the probabilities of reaching an equivalence class; it needs not to consider the labels of transitions and is different from our definitions on behavioral equivalences in π_N .

Because the union of equivalence relations may not be an equivalence relation, unlike in π , the union of reduction bisimulations is not a reduction bisimulation in general. Nevertheless, we have the following result.

Proposition 4.2. Let $\simeq = (\bigcup_i \mathcal{R}_i)^*$, where \mathcal{R}_i is a reduction bisimulation on *Proc*. Then \simeq is the largest reduction bisimulation on *Proc*.

Proof. It suffices to show that \simeq is a reduction bisimulation on $Proc$. Suppose that $(P, Q) \in \simeq$ and $P \xrightarrow{\tau} \eta_0$. Then there are $P_0, \dots, P_n \in Proc$ and reduction bisimulations $\mathcal{R}_1, \dots, \mathcal{R}_n$ such that $P = P_0$, $Q = P_n$, and $(P_{i-1}, P_i) \in \mathcal{R}_i$ for $i = 1, \dots, n$. By definition, there exist η_1, \dots, η_n such that $P_i \xrightarrow{\tau} \eta_i$ and $\eta_{i-1}(\tau, C_{i'}) = \eta_i(\tau, C_{i'})$ for $i = 1, \dots, n$ and any $C_{i'} \in Proc/\mathcal{R}_{i'}$. Note that for any $C \in Proc/\simeq$ and $\mathcal{R}_{i'}$, it follows from $\mathcal{R}_{i'} \subseteq \simeq$ that $C = \bigcup_j C_{i'j}$ for some $C_{i'j} \in Proc/\mathcal{R}_{i'}$, and moreover, $\bigcup_j C_{i'j}$ is a disjoint union since every $C_{i'j}$ is an equivalence class. We thus have that

$$\begin{aligned} \eta_{i-1}(\tau, C) &= \eta_{i-1}(\tau, \bigcup_j C_{i'j}) \\ &= \sum_j \eta_{i-1}(\tau, C_{i'j}) \\ &= \sum_j \eta_i(\tau, C_{i'j}) \\ &= \eta_i(\tau, C) \end{aligned}$$

for each $i = 1, \dots, n$. This means that $\eta_0(\tau, C) = \eta_n(\tau, C)$. Hence, \simeq is a reduction bisimulation, as desired. \square

The reduction bisimulation \simeq is called *reduction bisimilarity*. In other words, P and Q are *reduction bisimilar* if $(P, Q) \in \mathcal{R}$ for some reduction bisimulation \mathcal{R} .

As a process equivalence, reduction bisimilarity is seriously defective. For example, it relates any two processes that have no internal actions, such as $\bar{x}a$ and $\bar{y}a$. To obtain a satisfactory process equivalence, it is therefore necessary to allow more to be observed of processes. In the sequel, we use θ to range over barbs but τ . Based upon Lemma 3.9, we have the following definition.

Definition 4.3. The *observability predicate* \downarrow_θ in the π_N -calculus is defined as follows:

- (1) $P \downarrow_a$ if P has an input transition group with subject a .
- (2) $P \downarrow_{\bar{a}}$ if P has an output transition group with subject a .

Taking observability into consideration, we modify the notion of reduction bisimulation as follows.

Definition 4.4. An equivalence relation \mathcal{R} on $Proc$ is a *(strong) barbed bisimulation* if whenever $(P, Q) \in \mathcal{R}$,

- (1) $P \downarrow_\theta$ implies $Q \downarrow_\theta$;
- (2) $P \xrightarrow{\tau} \mu$ implies $Q \xrightarrow{\tau} \eta$ for some η satisfying $\mu(\tau, C) = \eta(\tau, C)$ for any $C \in Proc/\mathcal{R}$.

Analogous to Proposition 4.2, we have the following fact.

Proposition 4.5. Let $\sim = (\bigcup_i \mathcal{R}_i)^*$, where \mathcal{R}_i is a barbed bisimulation on $Proc$. Then \sim is the largest barbed bisimulation on $Proc$.

Proof. Since barbed bisimulations are reduction bisimulations, we see that \sim is a reduction bisimulation by Proposition 4.2. For any $(P, Q) \in \sim$, it is clear that $P \downarrow_\theta$ implies $Q \downarrow_\theta$. Thereby, \sim is a barbed bisimulation, and moreover, it is the largest one since it includes all barbed bisimulations on $Proc$. \square

The barbed bisimulation \sim is called *(strong) barbed bisimilarity*; we say that P and Q are *(strong) barbed bisimilar*, denoted $P \sim Q$, if $(P, Q) \in \mathcal{R}$ for some barbed bisimulation \mathcal{R} . It follows readily from definition that barbed bisimilarity is properly included in reduction bisimilarity. In addition, the fact below is also obvious.

Lemma 4.6. If $P \equiv Q$, then $P \sim Q$.

Like reduction bisimilarity, barbed bisimilarity is not satisfactory as a process equivalence as well. Nevertheless, it will underpin two good relations, barbed equivalence and barbed congruence. Let us begin with barbed equivalence.

Definition 4.7. Two processes P and Q in π_N are called *(strong) barbed equivalent*, denoted $P \approx Q$, if $P|R \sim Q|R$ for any R .

It follows directly from the above definition that $\approx \subseteq \sim$. On the other hand, there are a large number of counter-examples to show that $\sim \not\subseteq \approx$. For example, if $P \stackrel{\text{def}}{=} \bar{x}a.\bar{y}b$ and $Q \stackrel{\text{def}}{=} \bar{x}a$, then for any channel matrix of x , we have that $P \sim Q$ by definition. However, if one takes $R \stackrel{\text{def}}{=} x(w)$, then there exists $P|R \{\xrightarrow{\tau}_1 \bar{y}b\}$, and moreover, it cannot be matched by $Q|R$ because the unique transition group $Q|R \{\xrightarrow{\tau}_1 \mathbf{0}\}$ having τ as barb yields $\bar{y}b \not\sim \mathbf{0}$. Hence, $P|R \not\sim Q|R$ does not hold.

Further, we have the following fact that will be of use later.

Lemma 4.8.

(1) If $P \approx Q$, then $(\nu z)(P|R) \approx (\nu z)(Q|R)$ for any R and z .

(2) Let \mathcal{S} be an equivalence relation included in \sim . If for any R and z , $(P, Q) \in \mathcal{S}$ implies $((\nu z)(P|R), (\nu z)(Q|R)) \in \mathcal{S}$, then $\mathcal{S} \subseteq \approx$.

Proof. For (1), assume that $P \approx Q$. By definition, we see that $P|R \sim Q|R$ for any R . Moreover, it is straightforward to show that $(\nu z)P \approx (\nu z)Q$ for any z . Therefore, the assertion (1) holds.

For (2), suppose that $(P, Q) \in \mathcal{S}$. Then we see by the hypothesis of \mathcal{S} that for any R , $(P|R, Q|R) \in \mathcal{S} \subseteq \sim$. Hence, $P|R \sim Q|R$ for any R . So $P \approx Q$, and thus $\mathcal{S} \subseteq \approx$, finishing the proof. \square

Finally, we introduce the concept of barbed congruence.

Definition 4.9. Two processes P and Q in π_N are *(strong) barbed congruent*, denoted $P \simeq Q$, if $\mathcal{C}[P] \approx \mathcal{C}[Q]$ for every process context \mathcal{C} .

In other words, two terms are barbed congruent if the agents obtained by placing them into an arbitrary context are barbed bisimilar. The following remark clarifies the relationship between barbed equivalence and barbed congruence

Remark 4.10. By definition, we see that barbed congruent processes in the π_N -calculus are barbed equivalent, namely, $\simeq \subseteq \approx$. In fact, this inclusion is strict. The following example serves:

Let us take $P \stackrel{\text{def}}{=} x(w).[w = y]\tau$ and $Q \stackrel{\text{def}}{=} x(w).[w = z]\tau$, and suppose that all communication channels, except for x , are noiseless. We also assume that the channel matrix of x is given by

$$\begin{aligned} p_x(y|y) &= 0.5, & p_x(z|y) &= 0.5; \\ p_x(y|z) &= 0.5, & p_x(z|z) &= 0.5; \\ p_x(s|s) &= 1 \text{ for any } s \neq y, z. \end{aligned}$$

It follows readily that for any $R \in Proc$ with $R \xrightarrow{\alpha_i}_{p_i} R'_i$ for $i \in I$, if $\alpha_i \notin \{\bar{x}y, \bar{x}z, \bar{x}(y), \bar{x}(z)\}$, then $P|R \sim Q|R$. Moreover, if there exists $\alpha_i \in \{\bar{x}y, \bar{x}z, \bar{x}(y), \bar{x}(z)\}$, then any transition group of R having α_i as an action must be one of the following forms:

- (1) $R \{ \xrightarrow{\bar{x}y}_{0.5} R'_1 \} \uplus \{ \xrightarrow{\bar{x}z}_{0.5} R'_1 \};$
- (2) $R \{ \xrightarrow{\bar{x}(y)}_{0.5} R'_2 \} \uplus \{ \xrightarrow{\bar{x}z}_{0.5} (\nu y)R'_2 \};$
- (3) $R \{ \xrightarrow{\bar{x}y}_{0.5} (\nu z)R'_3 \} \uplus \{ \xrightarrow{\bar{x}(z)}_{0.5} R'_3 \};$
- (4) $R \{ \xrightarrow{\bar{x}(y)}_{0.5} (\nu z)R'_4 \} \uplus \{ \xrightarrow{\bar{x}(z)}_{0.5} (\nu y)R'_4 \}.$

For the form (1), we have that

$$\begin{aligned} P|R \{ \xrightarrow{\tau}_{0.5} \tau|R'_1 \} \uplus \{ \xrightarrow{\tau}_{0.5} R'_1 \} \text{ and} \\ Q|R \{ \xrightarrow{\tau}_{0.5} \tau|R'_1 \} \uplus \{ \xrightarrow{\tau}_{0.5} R'_1 \}, \end{aligned}$$

which means that $P|R \sim Q|R$. For the form (2), we have that

$$\begin{aligned} P|R \{ \xrightarrow{\tau}_{0.5} (\nu y)(\tau|R'_2) \} \uplus \{ \xrightarrow{\tau}_{0.5} (\nu y)R'_2 \} \text{ and} \\ Q|R \{ \xrightarrow{\tau}_{0.5} \tau|(\nu y)R'_2 \} \uplus \{ \xrightarrow{\tau}_{0.5} (\nu y)R'_2 \}. \end{aligned}$$

This yields that $P|R \sim Q|R$ because of $(\nu y)(\tau|R'_2) \equiv \tau|(\nu y)R'_2$. The form (3) is similar to that of the form (2), and we can also get that $P|R \sim Q|R$. For the form (4), we see that

$$\begin{aligned} P|R \{ \xrightarrow{\tau}_{0.5} (\nu y)(\tau|(\nu z)R'_4) \} \uplus \{ \xrightarrow{\tau}_{0.5} (\nu y, z)R'_4 \} \text{ and} \\ Q|R \{ \xrightarrow{\tau}_{0.5} (\nu z)(\tau|(\nu y)R'_4) \} \uplus \{ \xrightarrow{\tau}_{0.5} (\nu y, z)R'_4 \}. \end{aligned}$$

Because $(\nu y)(\tau|(\nu z)R'_4) \equiv \tau|(\nu y, z)R'_4 \equiv (\nu z)(\tau|(\nu y)R'_4)$, we get that $P|R \sim Q|R$. Summarily, we have that $P|R \sim Q|R$ for any R , and thus $P \approx Q$.

On the other hand, let $\mathcal{C} = x(y).[\mid \bar{x}s.\bar{x}s$. Then $\mathcal{C}[P] = x(y).x(w).[w = y]\tau|\bar{x}s.\bar{x}s$ and $\mathcal{C}[Q] = x(y).x(w).[w = z]\tau|\bar{x}s.\bar{x}s$. It is easy to check that $\mathcal{C}[P] \not\approx \mathcal{C}[Q]$, and thus $P \not\approx Q$ does not hold, as desired. \square

5 Bisimilarity

As mentioned in the last section, behavioral equivalences under a powerful observer that can differentiate between noisy free output and noisy bound output are investigated in this section.

We begin with a classical notion, bisimulation.

Definition 5.1. An equivalence relation \mathcal{R} on $Proc$ is a (strong) *bisimulation* if whenever $(P, Q) \in \mathcal{R}$, $P \xrightarrow{\mu}$ implies $Q \xrightarrow{\eta}$ for some η satisfying $\mu(\alpha, C) = \eta(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc/\mathcal{R}$.

The next result gives the largest bisimulation.

Proposition 5.2. Let $\sim = (\bigcup_i \mathcal{R}_i)^*$, where \mathcal{R}_i is a bisimulation on $Proc$. Then \sim is the largest bisimulation on $Proc$.

Proof. It is similar to that of Proposition 4.2. \square

The largest bisimulation \sim is called *(strong) bisimilarity*. In other words, P and Q are *(strong) bisimilar*, written $P \sim Q$, if $(P, Q) \in \mathcal{R}$ for some bisimulation \mathcal{R} . As an immediate consequence of Definitions 4.4 and 5.1, we have the following.

Lemma 5.3. Any two bisimilar processes are barbed bisimilar, i.e., $\sim \subseteq \dot{\sim}$.

The remark below tells us that bisimilarity is not included in barbed congruence.

Remark 5.4. Just like in the π -calculus, bisimilar processes in the π_N -calculus may not be barbed congruent, that is, $\sim \not\subseteq \dot{\sim}$. The following counter-example serves: Let $P \stackrel{\text{def}}{=} \bar{a}u|b(v)$ and $Q \stackrel{\text{def}}{=} \bar{a}u.b(v) + b(v).\bar{a}u$, and suppose, for simplicity, that there is a channel x with $p_x(b|b) = 1$. Then we see that $P \sim Q$. However, the context $\mathcal{C} \stackrel{\text{def}}{=} (x(a).[\])\bar{x}b$ yields that $\mathcal{C}[P] \not\dot{\sim} \mathcal{C}[Q]$. Therefore, $P \dot{\sim} Q$ does not hold. \square

The following notion will provide an equivalent characterization of bisimilarity.

Definition 5.5. A family of binary relations \sim_n on $Proc$ stratifying the bisimilarity is defined inductively as follows:

- (1) \sim_0 is the universal relation on processes.
- (2) For any $0 < n < \infty$, $(P, Q) \in \sim_n$ if
 - (2.1) $P \longrightarrow \mu$ implies $Q \longrightarrow \eta$ for some η satisfying $\mu(\alpha, C_{n-1}) = \eta(\alpha, C_{n-1})$ for any $\alpha \in Act$ and $C_{n-1} \in Proc / \sim_{n-1}$, and
 - (2.2) $Q \longrightarrow \eta$ implies $P \longrightarrow \mu$ for some μ satisfying $\mu(\alpha, C_{n-1}) = \eta(\alpha, C_{n-1})$ for any $\alpha \in Act$ and $C_{n-1} \in Proc / \sim_{n-1}$.
- (3) $(P, Q) \in \sim_\infty$ if $(P, Q) \in \sim_n$ for all $n < \infty$.

Observe that every \sim_n is an equivalence relation, so the notation $Proc / \sim_{n-1}$ in the above definition makes sense. Note also that $\sim_0, \sim_1, \dots, \sim_\infty$ is a decreasing sequence of relations.

The result below gives another way to check bisimilarity.

Proposition 5.6. $P \sim Q$ if and only if $P \sim_\infty Q$.

Proof. We first prove the necessity. By definition, we only need to show that $\sim \subseteq \sim_n$ for all $n < \infty$. Proceed by induction on n . The case $n = 0$ is trivial. Assume that $\sim \subseteq \sim_{n-1}$. For any $(P, Q) \in \sim$ and $P \longrightarrow \mu$, it follows from the definition of \sim that there exists η such that $Q \longrightarrow \eta$ and $\mu(\alpha, C) = \eta(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc / \sim$. By induction hypothesis $\sim \subseteq \sim_{n-1}$, we see that $\mu(\alpha, C_{n-1}) = \eta(\alpha, C_{n-1})$ for any $\alpha \in Act$ and $C_{n-1} \in Proc / \sim_{n-1}$. Consequently, $\sim \subseteq \sim_n$, as desired.

Now, let us show the sufficiency. It is enough to prove that \sim_∞ is a bisimulation. Suppose that $P \sim_\infty Q$ and $P \longrightarrow \mu$. Then for each $n < \infty$, there is η_n such that $Q \longrightarrow \eta_n$ and $\mu(\alpha, C_n) = \eta_n(\alpha, C_n)$ for any $\alpha \in Act$ and $C_n \in Proc / \sim_n$. By Lemma 3.10, the number of distinct η_n 's is finite, so there is η' such that $\eta' = \eta_n$ for infinitely many n . Since $\sim_0, \sim_1, \dots, \sim_\infty$ is a decreasing sequence of equivalence relations, we get that $\mu(\alpha, C_n) = \eta'(\alpha, C_n)$ for any $C_n \in Proc / \sim_n$. This gives rise to $(P, Q) \in \sim_n$ for all $n < \infty$, which means that $(P, Q) \in \sim_\infty$. Therefore, \sim_∞ is a bisimulation, finishing the proof. \square

For later need, let us pause to develop a “bisimulation up to” technique. Firstly, we make the following definition.

Definition 5.7. A binary symmetric relation \mathcal{R} on $Proc$ is a *bisimulation up to* \sim if whenever $(P, Q) \in \mathcal{R}$, $P \longrightarrow \mu$ implies $Q \longrightarrow \eta$ for some η satisfying $\mu(\alpha, C) = \eta(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc/(\mathcal{R} \cup \sim)^*$.

The following fact shows that any bisimulation up to \sim relation is a bisimulation, as expected.

Proposition 5.8. If \mathcal{R} is a bisimulation up to \sim , then $\mathcal{R} \subseteq \sim$.

Proof. Let $\mathcal{S} = (\mathcal{R} \cup \sim)^*$. For any $(P, Q) \in \mathcal{S}$, there exist P_0, \dots, P_n such that $P_0 = P$, $P_n = Q$, and $(P_{i-1}, P_i) \in \mathcal{R} \cup \sim$ for $i = 1, \dots, n$. If $(P_{i-1}, P_i) \in \mathcal{R}$, then $P_{i-1} \longrightarrow \mu_{i-1}$ implies $P_i \longrightarrow \mu_i$ for some μ_i satisfying $\mu_{i-1}(\alpha, C) = \mu_i(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc/\mathcal{S}$. If $(P_{i-1}, P_i) \in \sim$, then $P_{i-1} \longrightarrow \mu_{i-1}$ implies $P_i \longrightarrow \mu_i$ for some μ_i satisfying $\mu_{i-1}(\alpha, C') = \mu_i(\alpha, C')$ for any $\alpha \in Act$ and $C' \in Proc/\sim$. This, together with the fact $\sim \subseteq \mathcal{S}$, yields that $\mu_{i-1}(\alpha, C) = \mu_i(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc/\mathcal{S}$. As a result, for any $P_0 \longrightarrow \mu_0$, there are μ_1, \dots, μ_n such that for $i = 1, \dots, n$, $P_i \longrightarrow \mu_i$ and $\mu_{i-1}(\alpha, C) = \mu_i(\alpha, C)$ for any $\alpha \in Act$ and $C \in Proc/\mathcal{S}$. Therefore, \mathcal{S} is a bisimulation, and thus $\mathcal{R} \subseteq \sim$. This completes the proof. \square

We now establish a bisimulation up to \sim relation which will be used in the next section.

Lemma 5.9. Let

$$\mathcal{R} = \{((\nu \tilde{z})(P|R), (\nu \tilde{z})(Q|R)) : \tilde{z} \text{ are arbitrary names, and } P, Q, R \in Proc \text{ with } P \sim Q\}.$$

Then \mathcal{R} is a bisimulation up to \sim .

Proof. Clearly, \mathcal{R} is an equivalence relation. To check that it is a bisimulation up to \sim , we appeal to a case analysis on the last rules applied in the inference of related transitions. It needs to examine all inductive steps through combinations of the transition group rules of Par, Comm, Res-Out, Res-Inp, Res-Tau, and Open-Inp. This is a long and routine argument, so we omit the details here. \square

To obtain a congruence based on actions, we make the following definition.

Definition 5.10. Two processes P and Q in π_N are called (*strong*) *full bisimilar*, denoted $P \simeq Q$, if $P\sigma \sim Q\sigma$ for any substitution σ .

Like in π , bisimilarity in π_N is not preserved by substitution, as illustrated below.

Remark 5.11. It follows immediately from definition that full bisimilar processes are necessarily bisimilar, i.e., $\simeq \subseteq \sim$. Nevertheless, $\sim \not\subseteq \simeq$. In other words, there are some bisimilar processes that are not full bisimilar. For example, let $P \stackrel{\text{def}}{=} \bar{a}u|b(v)$ and $Q \stackrel{\text{def}}{=} \bar{a}u.b(v) + b(v).\bar{a}u$. Then for any noisy channels, we always have that $P \sim Q$. However, the substitution σ defined by

$$\sigma(x) = \begin{cases} a, & \text{if } x = b \\ x, & \text{otherwise} \end{cases}$$

gives rise to that $P\sigma = \bar{a}u|a(v)$ and $Q\sigma = \bar{a}u.a(v) + a(v).\bar{a}u$. Obviously, $P\sigma \not\sim Q\sigma$, and thus P and Q are not full bisimilar. \square

6 A hierarchy of behavioral equivalences

In the previous two sections, we have introduced several behavioral equivalences. Some simple inclusion relationships among them have been established. In this section, we consummate the relationships and then give a hierarchy of these behavioral equivalences.

The following theorem shows that bisimilar processes are barbed equivalent.

Theorem 6.1. For any two processes P and Q in π_N , if $P \sim Q$, then $P \dot{\sim} Q$.

Proof. By Lemmas 5.8 and 5.9, we see that $P \sim Q$ implies $(\nu z)(P|R) \sim (\nu z)(Q|R)$ for any R and z . Using the fact $\sim \subseteq \dot{\sim}$ obtained in Lemma 5.3, we get from Lemma 4.8 (2) that $\sim \subseteq \dot{\sim}$, thus finishing the proof. \square

Remark 6.2. In the π -calculus, it is well known that strong barbed equivalence coincides with strong bisimilarity; see, for example, Theorem 2.2.9 in [43]. However, in the π_N -calculus the converse of the above theorem is not true in general. In other words, two barbed equivalent processes may not be bisimilar. For example, let us consider the barbed equivalent processes $P \stackrel{\text{def}}{=} x(w).[w = y]\tau$ and $Q \stackrel{\text{def}}{=} x(w).[w = z]\tau$ in Remark 4.10, and keep the hypothesis of related channel matrices. Then we find that

$$P\{\xrightarrow{xy}_1 \tau\} \text{ and } Q\{\xrightarrow{xy}_1 \mathbf{0}\}.$$

This shows us that $P \not\sim Q$. Obviously, this non-coincidence of barbed equivalence and bisimilarity arises from the noise of channel x . \square

We continue to discuss the relationship between barbed congruence and full bisimilarity. To this end, we need one more concept.

Definition 6.3. An equivalence relation \mathcal{R} on processes is said to be a *process congruence* if $(P, Q) \in \mathcal{R}$ implies $(\mathcal{C}[P], \mathcal{C}[Q]) \in \mathcal{R}$ for every process context \mathcal{C} .

The following is an easy consequence, which is useful for checking process congruence.

Proposition 6.4. An equivalence relation \mathcal{R} is a process congruence if and only if it is preserved by all elementary contexts.

The next observation gives a basic process congruence; its proof follows immediately from the definition of structural congruence.

Proposition 6.5. The structural congruence \equiv is a process congruence.

As an immediate consequence of Definition 4.9, Proposition 6.5, and Lemma 4.6, we have the following.

Corollary 6.6. If $P \equiv Q$, then $P \dot{\sim} Q$.

For subsequent need, we show that \simeq is also a process congruence.

Lemma 6.7. \simeq is a process congruence, and moreover, it is the largest process congruence included in \sim .

Proof. We first show that \simeq is a process congruence. Suppose that $P \simeq Q$, i.e., $P\sigma \sim Q\sigma$ for every substitution σ . By Proposition 6.4, we only need to prove that for every elementary context \mathcal{C} and any substitution σ , $\mathcal{C}[P\sigma] \sim \mathcal{C}[Q\sigma]$. For $\mathcal{C} = \pi.[\] + M$, if the prefix π is of form $\bar{x}y$, τ , or $[x = y]\pi$, then it follows directly from $P\sigma \sim Q\sigma$ that $\mathcal{C}[P\sigma] \sim \mathcal{C}[Q\sigma]$. In the case $\mathcal{C} = x(z).[\] + M$, we see that $\mathcal{C}[P\sigma] = x(z).P\sigma + M$ and $\mathcal{C}[Q\sigma] = x(z).Q\sigma + M$. Because z is not strongly bound in $P\sigma$ or $Q\sigma$, we get by $P \simeq Q$ that for any z' , $P\sigma\{z'/z\} \sim Q\sigma\{z'/z\}$. This gives rise to $x(z).P\sigma \sim x(z).Q\sigma$, and thus $\mathcal{C}[P\sigma] \sim \mathcal{C}[Q\sigma]$. By carrying out an analysis of the transition group rules related to composition, restriction, and replication, it is routine to check that $\mathcal{C}[P\sigma] \sim \mathcal{C}[Q\sigma]$ holds for the other four elementary contexts, and we do not go into the details.

By definition, we see that $\simeq \subseteq \sim$. We now verify that \simeq is the largest process congruence included in \sim . Let \sim^* be an arbitrary process congruence included in \sim . Suppose that $P \sim^* Q$ and $\sigma = \{y_1, \dots, y_n/x_1, \dots, x_n\}$. Without loss of generality, we assume that there is a noiseless channel s with $s \notin \text{fn}^*(P\sigma, Q\sigma)$, and set

$$\mathcal{C} \stackrel{\text{def}}{=} (\nu s)(\bar{s}y_1 \dots \bar{s}y_n | s(x_1) \dots s(x_n).[\]).$$

Then $\mathcal{C}[P] \sim^* \mathcal{C}[Q]$, hence $\mathcal{C}[P] \sim \mathcal{C}[Q]$. Notice that $\mathcal{C}[P]\{(\xrightarrow{1}^\tau)^n(\nu s)P\sigma \equiv P\sigma\}$, where $(\xrightarrow{1}^\tau)^n$ is the n -fold composition of $\xrightarrow{1}^\tau$, therefore $\mathcal{C}[Q]\{(\xrightarrow{1}^\tau)^n \sim P\sigma\}$. But $\mathcal{C}[Q]\{(\xrightarrow{1}^\tau)^n(\nu s)Q\sigma \equiv Q\sigma\}$ only, so $P\sigma \sim Q\sigma$. We thus get that $P \simeq Q$, and hence $\sim^* \subseteq \simeq$, as desired. \square

Based on the previous lemmas, we can prove the next result.

Theorem 6.8. For any two processes P and Q in π_N , if $P \simeq Q$, then $P \dot{\simeq} Q$.

Proof. We see from Lemma 6.7 that \simeq is a process congruence included in \sim . Since $\sim \subseteq \dot{\sim}$ by Theorem 6.1, \simeq is a process congruence included in $\dot{\sim}$. By definition, $\dot{\simeq}$ is the largest process congruence included in $\dot{\sim}$, therefore we have that $\simeq \subseteq \dot{\simeq}$, finishing the proof of the theorem. \square

It would be expected that $\dot{\simeq} \subseteq \simeq$. However, this is not true, as we shall see.

Remark 6.9. Like Theorem 6.1, the converse of the above theorem is not true in general, that is, two barbed congruent processes may not be full bisimilar. Even two barbed congruent processes may not be bisimilar in the π_N -calculus. For example, take $P \stackrel{\text{def}}{=} (\nu y)\bar{x}y.\bar{x}y$ and $Q \stackrel{\text{def}}{=} (\nu y)\bar{x}y.(\nu y)\bar{x}y$, and suppose, for simplicity, that the channel x is noiseless. It is easy to check by induction on context \mathcal{C} that $\mathcal{C}[P] \dot{\sim} \mathcal{C}[Q]$ holds for any context. Consequently, $P \dot{\simeq} Q$ by definition. Nevertheless, notice that there is a transition group $P\{\xrightarrow{1}^{\bar{x}(y)} \bar{x}y\}$ and the only transition group of Q making $P \sim Q$ possible is $Q\{\xrightarrow{1}^{\bar{x}(y)} (\nu y)\bar{x}y\}$. But it is obvious that $\bar{x}y \not\sim (\nu y)\bar{x}y$. Hence, $P \not\sim Q$. \square

Finally, based on our results in Sections 4–6, we summarize the hierarchy of the behavioral equivalences in the π_N -calculus, which is depicted in Figure 1(b).

Theorem 6.10. In the π_N -calculus,

- (1) $\simeq \subseteq \dot{\simeq} \subseteq \dot{\sim} \subseteq \sim \subseteq \dot{\sim}$ and $\simeq \subseteq \dot{\sim} \subseteq \dot{\sim}$; each of the inclusions can be strict.
- (2) Neither $\sim \subseteq \dot{\simeq}$ nor $\dot{\simeq} \subseteq \sim$ holds.

7 Conclusion

This paper is devoted to a hierarchy of behavioral equivalences in the π_N -calculus, the π -calculus with noisy channels. First, we have developed an early transitional semantics of the π_N -calculus and provided two presentations of the transition rules. It is worth noting that this semantics is not a directly translated version of the late semantics of π_N in [54], and we have found that not all bound names are compatible with alpha-conversion in the noisy environment, which is a striking dissimilarity between π_N and π . As a result, an Open-Inp rule for inputting bound names is required. Then we have introduced some notions of behavioral equivalences in π_N , including reduction bisimilarity, barbed bisimilarity, barbed equivalence, barbed congruence, bisimilarity, and full bisimilarity. Some basic properties of them have also been stated. Finally, we have established an integrated hierarchy of these behavioral equivalences. In particular, because of the noisy nature of channels, the coincidence of bisimilarity and barbed equivalence, as well as the coincidence of full bisimilarity and barbed congruence, in the π -calculus does not hold in π_N .

There are some limits and problems arising from the present work which are worth further studying. We only present a hierarchy of strong behavioral equivalences; the corresponding weak version that ignores invisible internal actions is a research topic. Some algebraic laws and axiomatizations of these behavioral equivalences are interesting problems for future research. A hierarchy of behavioral equivalences in some subcalculus (for example, asynchronous π -calculus where asynchronous observers are less discriminating than synchronous observers [7, 13, 17, 18, 24]) is yet to be addressed. Note that the converse statements of Theorems 6.1 and 6.8 cannot hold. Hence, a necessary and sufficient condition for the converse statements to be true is desirable. Finally, the reliability of processes in the π_N -calculus initiated in [54] remains an interesting issue when using the early transitional semantics of π_N and non-approximate bisimilarity.

Acknowledgment

The author would like to thank Professor Mingsheng Ying for some helpful discussions and invaluable suggestions.

References

- [1] P. A. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In R. Cleaveland, editor, *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 1999.
- [2] P. A. Abdulla, N. Bertrand, A. M. Rabinovich, and P. Schnoebelen. Verification of probabilistic systems with faulty communication. *Inf. Comput.*, 202(2):141–165, 2005.
- [3] P. A. Abdulla, L. Boasson, and A. Bouajjani. Effective lossy queue languages. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 639–651. Springer, 2001.

- [4] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In A. J. Hu and M. Y. Vardi, editors, *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 1998.
- [5] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Inf. Comput.*, 130(1):71–90, 1996.
- [6] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inf. Comput.*, 127(2):91–101, 1996.
- [7] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π -calculus. *Theor. Comput. Sci.*, 195(2):291–324, 1998.
- [8] M. Berger. Basic theory of reduction congruence for two timed asynchronous π -calculi. In P. Gardner and N. Yoshida, editors, *CONCUR'04*, volume 3170 of *Lecture Notes in Computer Science*, pages 115–130, London, UK, 2004. Springer.
- [9] M. Berger and K. Honda. The two-phase commitment protocol in an extended pi-calculus. In L. Aceto and B. Victor, editors, *Proceedings of EXPRESS '00*, volume 39 (1) of *ENTCS*. Elsevier Science, Amsterdam-Lausanne-New York-Oxford-Shannon-Tokyo, 2000.
- [10] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37(1):77–121, 1985.
- [11] M. Boreale and R. D. Nicola. Testing equivalence for mobile processes. *Inf. Comput.*, 120(2):279–303, 1995.
- [12] M. Boreale and D. Sangiorgi. Some congruence properties for π -calculus bisimilarities. *Theor. Comput. Sci.*, 198(1-2):159–176, 1998.
- [13] G. Boudol. Asynchrony and the π -calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis, May 1992.
- [14] P. Buchholz. Exact and ordinary lumpability in finite markov chains. *J. Appl. Prob.*, 31:59–75, 1994.
- [15] Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. *Theor. Comput. Sci.*, 373(1-2):92–114, 2007.
- [16] U. Engberg and M. Nielsen. A calculus of communicating systems with label-passing. Technical Report DAIMI PB-208, Comp. Sci. Department, Univ. of Aarhus, Denmark, 1986.
- [17] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*, pages 844–855. Springer, 1998.
- [18] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. *J. Log. Algebr. Program.*, 63(1):131–173, 2005.
- [19] O. M. Herescu. *The Probabilistic Asynchronous Pi-Calculus*. Dissertation, Pennsylvania State University, Dec. 2002.
- [20] O. M. Herescu and C. Palamidessi. Probabilistic asynchronous π -calculus. In J. Tiuryn, editor, *Foundations of Software Science and Computation Structures, Third International Conference, FOSSACS 2000, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*, volume 1784 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2000.

- [21] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, Cambridge, 1996.
- [22] C. A. R. Hoare. Communicating sequential processes. *Comm. ACM*, 21(8):666–677, 1978.
- [23] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
- [24] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In P. America, editor, *Proceedings of ECOOP '91*, volume 512 of *LNCS*, pages 133–147. Springer, July 1991.
- [25] S. P. Iyer and M. Narasimha. Probabilistic lossy channel systems. In M. Bidoit and M. Dauchet, editors, *TAPSOFT'97: Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France, April 14-18, 1997, Proceedings*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997.
- [26] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94:1–28, 1991.
- [27] R. Q. Lu and Z. C. Wei. Truly probabilistic pi-calculus and risk semantics. Technical report, Mathematical Institute, Academia Sinica, 2004.
- [28] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1980.
- [29] R. Milner. *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
- [30] R. Milner. The polyadic π -calculus: A tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, volume 94 of *Series F. NATO ASI*, Springer, 1993. Available as Technical Report ECS-LFCS-91-180, University of Edinburgh, October 1991.
- [31] R. Milner. *Communicating and Mobile Systems: the π -Calculus*. Cambridge University Press, Cambridge, May 1999.
- [32] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Inf. Comput.*, 100:1–77, 1992.
- [33] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theor. Comput. Sci.*, 114(1):149–171, 1993.
- [34] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Proceedings of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.
- [35] J. Parrow. An introduction to the π -calculus. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 479–543. Elsevier Science, Amsterdam-Lausanne-New York-Oxford-Shannon-Tokyo, 2001.
- [36] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, Bonn, Germany, 1962. (In German).
- [37] B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.
- [38] C. Priami. Stochastic π -calculus. *Comput. J.*, 38(7):578–589, 1995.
- [39] W. Reisig. *Petri Nets: An Introduction*. Springer-Verlag, New York, 1985.
- [40] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, Department of Computer Science, University of Edinburgh, 1992.
- [41] D. Sangiorgi. A theory of bisimulation for the π -calculus. *Acta Inform.*, 33(1):69–97, 1996.

- [42] D. Sangiorgi and D. Walker. On barbed equivalences in π -calculus. In K. G. Larsen and M. Nielsen, editors, *CONCUR 2001 - Concurrency Theory, 12th International Conference, Aalborg, Denmark, August 20-25, 2001, Proceedings*, volume 2154 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2001.
- [43] D. Sangiorgi and D. Walker. *The π -calculus: A Theory of Mobile Processes*. Cambridge University Press, Cambridge, 2001.
- [44] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. Comput.*, 2(2):250–273, 1995.
- [45] C. E. Shannon. A mathematical theory of communication, I, II. *Bell Syst. Techn. J.*, 27:379–423, 623–656, 1948.
- [46] P. Sylvain and C. Palamidessi. Expressiveness of probabilistic π -calculus. *Electr. Notes Theor. Comput. Sci.*, 164(3):119–136, 2006.
- [47] R. van Glabbeek, S. A. Smolka, B. Steffen, and C. M. N. Tofts. Reactive, generative, and stratified models of probabilistic processes. In J. C. Mitchell, editor, *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science*, pages 130–141, Philadelphia, PA, June 1990. IEEE Computer Society Press.
- [48] R. J. van Glabbeek, S. A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Inf. Comput.*, 121(1):59–80, 1995.
- [49] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.
- [50] M. S. Ying. *Topology of Process Calculus: Approximate Correctness and Infinite Evolution of Concurrent*. Springer-Verlag, New York, 2001.
- [51] M. S. Ying. Additive models of probabilistic processes. *Theor. Comput. Sci.*, 275(1-2):481–519, 2002.
- [52] M. S. Ying. Bisimulation indexes and their applications. *Theor. Comput. Sci.*, 275(1-2):1–68, 2002.
- [53] M. S. Ying. Reasoning about probabilistic sequential programs in a probabilistic logic. *Acta Inform.*, 39:315–389, 2003.
- [54] M. S. Ying. π -calculus with noisy channels. *Acta Inform.*, 41:525–593, 2005.